

Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

HARDWARE DAVID ANEAR
SOFTWARE JEFF RAE
FORTH DAVID WILSON
AMATEUR RADIO ROD DRYSDALE VK3BYU
EDUCATION JEFF KERRY
LIBRARY RON KERRY
TAPE LIBRARY JOHN WHITEHEAD
DISK LIBRARY WARREN SCHAECH (B.H.)
NEWSLETTER IAN EYLES
SYM. BRIAN CAMPBELL
SECRETARY ROSEMARY EYLES

OSI SYM KIM AIM UK101 RABBLE 65

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.4 No.3 December 1983



SEASONS GREETINGS

INDEX

Adventuring pt 1	4	Machine Language pt 18	3
BASIC Quirk	14	Meeting KAOS	2
EPROM Programmer - Listing	14	Meeting WA	2
EPROM Programmer 3	12	OSI History	6
Expansion Socket - 40 pin	13	Ratbas	7
Ext Mon - Mods	11	ROM BASIC - Correction	10
For Sale	15	SUPERBOARD	4
Front Page	2	Tape Viewer	15

FRONT PAGE

As usual, there will be no meeting in December, the next meeting will be on the 29th January 1984.

Please remember that membership fees are now due and if you have not paid by the end of January you will not receive the newsletter in February.

The closing date for articles for the January newsletter will be the 13th January.

We would like to wish you all a merry Christmas and a happy and prosperous New Year.



THE MEETING WAS KAOS

by King Corky

Well, the BBQ was great, everybody enjoyed themselves and I didn't see one 'tipsy' person there, although I was a bit suspicious about that chap in the yellow cap! Many thanks Rosemary and Ian for organising the 'DO'.

David Anear has designed a new Eprom burner to run from the 16-pin I/O bus. It has two Zero Insertion sockets, one each for the ROM and header. Ray Gardiner announced that the 8.1 monitor is not yet ready for release but they have updated their disassembler to include the new 65C02. Ray also demo'd his multi-window screen formats running under Multi-Tasking Forth in 80 columns. Looks fantastic.

From Paul Dodd and COMP-SOFT, they will be releasing a new BASIC Interpreter soon as well as a CP/M board that will plug straight into the OSI and Rabble via the 16-pin I/O. It will have on-board, 64K RAM and a floppy controller. Michael Lemaire is working on the Roland Synthesizer interface for the Rabble, OSI, Apple and ATARI. Should be available for demo next meeting. Ask Michael about 'RATBAS'. I can't remember if it means Ratified BASIC or something else. My hearing was even worse than usual that day.

Nothing much else to say except MERRY CHRISTMAS to you all and may you write many fine programs, before the next meeting, that you can bring along and show us.



KAOS-WA MEETING

Our last meeting saw only 7 members (no new ones this time) and 5 computers (4 working). Wayne Geary demonstrated his graphics program, voice generator (used to announce the start of the meeting) and a flight simulator program. Peter van der Weddon had his 8" disks working and demonstrated some of the software he has acquired for it. Most of the meeting consisted of members discussing problems they had come across since our last meeting and different ways to overcome them.

The next meeting (at which we will be ONE) will be on Sunday 15th of January 1984 at 2.00pm. The venue will be our usual meeting place - top floor GUILD HOUSE, 56 Kishorn Rd, Mt Pleasant.
Gerry Ligtermoet,

This month in BMLP a new addressing mode:-Accumulator addressing.

As its name implies, Accumulator addressing defines the operand of the instruction as the Accumulator.

Accumulator mode instructions are all single byte instructions. In this respect they appear to have similar properties to inherent or implied mode instructions.

The essential difference is that inherent mode instructions are restricted to inherent mode whilst instructions that utilise accumulator mode can also make use of other addressing modes. i.e Instructions available in inherent mode apply only to the internal registers of the 6502; Instructions available in Accumulator mode however are also available in other addressing modes which allow them to operate on memory.

There are only four instructions in the entire 6502 instruction set which utilise the accumulator mode but these provide some very powerful and useful programming tools. Next month we will start to examine these instructions.

For the moment let's take a look back over the last 17 issues of the KAOS magazine.

The BMLP series of articles takes up about 1/8th of the article space over that time; nearly 2 pages per month.

A couple of times I nearly didn't make the publishing deadline, like this month for example. If I hadn't, it is possible that some of those pages would have been blank. One thing is sure, Ian Eyles would have had to scratch real deep to fill them with something else.

Next March, three short months from now, BMLP will have run its course and there IS going to be 2 blank pages to fill: EVERY MONTH.
We need articles and we need them NOW!

Take another look over your back issues of KAOS. How many names do you see on the articles there? Two maybe three dozen?
What are the other 300 of you doing?
Maybe you think that no-one else would be interested in your pet project?
Perhaps you find the prospect of going into print a little daunting? I know I did.

OR IS IT JUST TOO MUCH TROUBLE?

Just think, if we all felt like that there wouldn't be a KAOS. We could all join the VIC 20 group or go back to watching TV at night.

Now I'm not going to ask you to become a regular KAOS author. But, if we all did just one small paragraph A YEAR we would never be wanting for articles. One paragraph, maybe half an hours work is that too much to ask?

The OSI community has always been short on information. Today more than ever we need to share what we have no matter how small! The information gap is still with us!

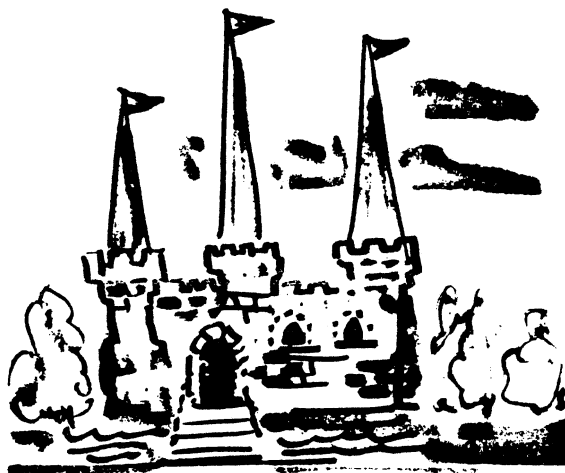
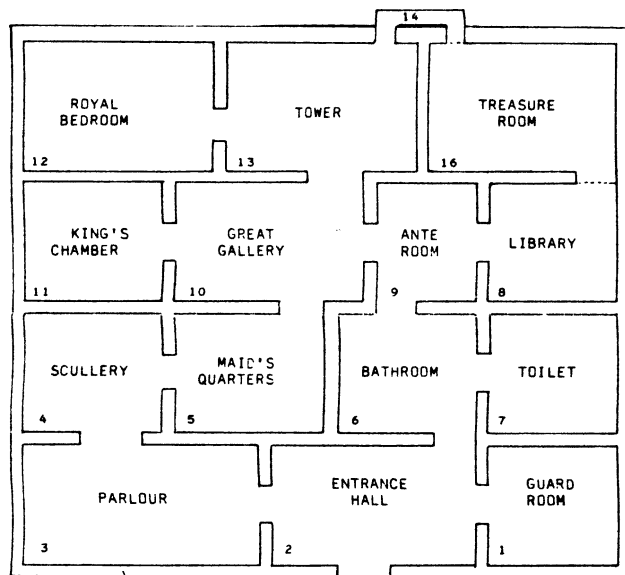
Make it a New Years Resolution.....write for KAOS.

David Dodds

Superboard

December, 1983

NEWSLETTER OF THE OHIO SUPERBOARD USER GROUP, 146 YORK STREET, NUNDAH, 4012.



ADVENTURING - PART ONE

Last month's very simple dungeon game may have given you a taste for real adventuring. Adventures are the most difficult form of puzzles, and the more advanced ones can take many hours to solve, and even longer to think out and write.

This month we will disassemble the Treasure Quest program, explaining the variables and subroutines, and next month there will be an article on the writing of more complex adventures.

Above you will see a floor plan of our castle. Get out KAOS 4/2 and find the Treasure Quest listing.

In line 100, CHR\$(26) is a Cegmon screen clear, and the pokes set up a get-key routine, called with USR(X).

Line 110 sets up all the major variables. In order these are:-

- N , number of rooms. M , men killed in the Quest for the treasure.
- F\$(X,Y) , fight sequences, X being one of the 5 scenarios, and Y being the particular part of the 5 step sequence that is appropriate.(DATA 550)
- W\$(X) , the names of the 5 weapons. The correct weapon needs to be in hand to defeat the hazard encountered if a fight ensues. (DATA 660)
- R\$(N) , the names of the 16 rooms. (DATA 670)
- D\$(4) , the names of the four directions of movement. (DATA 710)
- C(N,4) , the room connections. Refer to the floor plan and DATA lines 720 to 740. 0 denotes no doorway, and -1 means outside the castle.

The Data is in groups of four, starting at room 1, which is the Guard Room. Our Hero starts in the Entrance Hall which is room 2, so look at the second sequence of 4's, and you will see that North is room 6 (Bathroom), East is the Guard room (1), South is outside (-1), and West is the Parlour (3).

The Treasure room has two one-way doors, shown dotted on the plan. Having entered by the secret passage, you can only exit via the Library.

— SUPERBOARD —

As a matter of interest, you could also exit in any other direction to, say, the Scullery, just by inserting the appropriate number in the data. This would make it rather difficult to draw a floor plan or even to visualise. With C(N,6), and 6 data bits per room, you could go up or down, and C(N,8) would get you into the 4th dimension!

- H(5) , the rooms where hazards exist. If H(1) was 6, the Guard would be in the bathroom.
 - A(3) , the arms carried by the Hero. He can only carry 3 at any time.
 - W(5) , the rooms where the five weapons are located. Arms = held weapons.
- Lines 120 to 140 read in the strings and variables from the data.
Line 150 puts weapons randomly in rooms 1 to 6, and hazards in rooms 8 to 13.
Line 160 puts the treasure, T in room 16, and starts the hero, H in room 2.
The hero is given a weapon chosen at random. W(R) removes the weapon selected from the room in which it was previously stored. Q is for Quest. If Q=2 then you have the treasure with you.

Lines 170 - 240 are the program loop. 170 checks if you have gone outside with or without the treasure. 190 finds out if you have gone north from the secret passage and fallen into the moat.
In line 210 we find the fight flag, F. If F=1, you have encountered one of the hazards and run. 340 is the fight subroutine.
Line 220. The weapons check/find/drop subroutine is 460. 280 is the movement subroutine.
Line 230 moves a hazard at random to rooms 1 to 13 if it is still alive.
This is to make the game a little less tame. You can't rely on the hazard still being there if you nip back to pick up the right weapon.
Lines 250 and 270 are the escape possibilities.

Lines 280 to 330 are the move routine. USR(X) calls the get-key routine at \$FD00, looking for N,E,S,W. 320 could have been IF PEEK(531)=ASC(D\$(R)), as the ASC function comes up with the ascii number of the leftmost character in the string.

Lines 340 to 450 is the fight subroutine. 370 gives a 50/50 chance of a fight. 400 moves the Hero to another room if he runs. In 430, if the Hero has the right arm (weapon), he kills the hazard. Subroutine 540 is a simple time delay to allow you time to read the screen, and give suspense.

Lines 460 to 490 is a Arms printout, and search of the room for more.
500 to 530 is the find sequence. If you are already holding 3 arms, and pick up another, you drop one. You can always return to the room to get it again if you need it.

Well, you now know everything about the 2.6K game and should be able to design a new floor plan, or change things around to your liking. Let's block off the access to the Tower from the Great Gallery, and make it accessible via the King's Chamber and Royal Bedroom. The Data starts on line 730 in room 8. Mark off every 4th comma. The first two blocks of 4 are rooms 8 and 9 - no change. From there, the new data will be :-

0,9,5,11,12,10, 0,0,0,13,11,0,14,0,0,12

The changes, underlined, read:- No door from room 10 northwards, a north door from room 11 to 12, a south door from room 12 to 11, and no south door in the Tower.

Real Adventures don't have any random routines, and are purely problems of logic, luck playing no part in the solution. If you would like to try a few adventures, then the 4 offered by Victory Software in "OSI GREATEST HITS VOLUME 2" at the special price to club members (see past KAOS articles), would be an excellent place to begin.

Ed Richardson.

OSI HISTORY
by Eric Lindsay

I recently saw a history of the Apple computer and being a dedicated OSI user, I decided to try and write a history of the OSI company. Information has been hard to come by but over the next couple of months KAOS will be printing the material I have managed to collect. If anyone can help by adding to or correcting this material, I would be grateful.

Ohio Scientific Instruments began operations in Hiram, Ohio around 1975 as one of the typical basement enterprises that characterised early operations in the microcomputer field. The Altair bus, later known as the S100 bus, had not as yet dominated the bus based microcomputer market, and South West Technical Products (SWTPC) were a major force in the micro world, with their 6800 processor based kits. The major market was hobbyists, with some businesses seeing that they could replace exceedingly expensive mini computers with the new micros.

Radio Shack were selling their 4k and 16k TRS80 Model 1, and talking about upgrading to the 12k Microsoft Basic. Apple were slowly expanding from yet another basement. IBM saw no threat in the new microprocessors, and certainly were not about to give micros their seal of approval by actually selling one of them. It was a time of small companies, of hasty design, of co-operative and well informed hobbyist groups. And it looked like a way of making a little profit, if you were a bit ahead of your competitors, or were cheaper.

Ohio Scientific were in the business of making add-on boards for various equipment. When Mike Cheiky, vice-president of OSI, and apparently designer of much of their equipment, did the 420 memory board he followed this cheap-is-better philosophy to the end. You can save costs by making boards without sockets, and since removing chips from a double sided board is difficult, why not save a bit more by making it with foil on one side only. This made for some fairly strange address line layouts when the OSI bus was later designed to suit the board. Most companies designed their bus first, and their boards later. This sort of decision was to prove typical of OSI.

The 48 pin system bus that resulted was also done on the cheap. It used four 12 pin molex connectors, which were fine for single sided boards, were cheap, and avoided the expense of gold plating. It wasn't the most reliable system of connectors, but as anyone who has had problems with board connectors can tell you, there probably isn't any such thing as a totally reliable board connector. The bus was advanced for its time. It includes 20 address lines, which allow a maximum user memory of one megabyte, and has an 8 bit bi-directional data bus. The S100 was still using two different data buses. Six control lines are available, and 6 power lines carry regulated voltages, which avoids the need for expensive, heat producing regulators on the actual boards. The power lines are arranged so that if a board is inserted the wrong way round, the power supplies are shorted out, thus causing them to shut down (at least, they do if you designed short circuit protection into them).

OSI probably had thoughts of making all its equipment compatible with that of SWTPC, but with the eclipse of that company, found itself one of the very few makers of bus based equipment not using the S100 bus. Rather than join the several hundred S100 manufacturers that then existed, it made a typical OSI decision, and started producing full computer systems using its own bus. This meant that it had to produce the same range of products as all the various S100 manufacturers. As a result, it ended up producing one of the largest range of microcomputers ever seen from one company.

In producing this range, it managed a variety of firsts. It was probably the first company to actually deliver completely assembled and tested

microcomputers, rather than kits. One of the first to provide Microsoft BASIC in ROM, and the first microcomputer company to have production model equipment with Winchester hard discs, way back in 1978.

Products ranged from 4k cassette based bare board systems for hobbyists, through advanced hobbyist mini floppy systems, to 56k 8" floppy or hard disc systems with multi-user operating systems, and multiple processors. Systems ran video displays that ranged from 24 by 24, up to 64 by 32, with 16 colors available. There were serial terminal systems, sound output as standard, voice input experimenter boards, joysticks, BSR home appliance controller interfaces, and a lot more. Whatever you could think of, OSI either had it for sale, were demonstrating it, or were planning to add it.

Continued next month



OF RATFOR AND RATBAS AND THE JOYS OF BOTH

by Michael Lemaire

Fortran is a reasonably primeval computer language which is still used rather a lot today. Fortran was the first "Second Generation" language written for computers, being put together by John Backus of IBM in the 1950s. Backus went on to serve on the international committees that developed Algol, and he invented "Backus Normal Form" (later "Backus-Naur Form") which is used to describe languages. Nowadays he is researching functional programming techniques.

Fortran, being the first attempt at improving on assembler language (First Generation), borrowed many of its features from the Von Neumann machine architecture; for instance the only control structure was a jump. Many programmers were enjoying the use of Fortran as a welcome break from assembler, but gradually, as new programming concepts evolved, people found themselves wishing for extra features in the language. One of these people was Dennis Ritchie. He wrote a program which processed an input program written in an extended version of Fortran, with "if..else", "begin..end", "while" and "repeat" (and more), and output a standard Fortran program, which could then be compiled. The program was called "Ratfor", for "Rational Fortran". It made Fortran much nicer to use. (Nice constructs such as "if..else..endif" were added in the 1977 Fortran Standard, but this was 11 years later.) It is interesting to note that many of the features of Ratfor programs look like C language constructs -- this is rather obvious really, when you discover that Dennis Ritchie and Brian Kernighan wrote C at a later date.

The idea of a preprocessor is to take a program written in an extended version of a language, and to convert it to a program written in a standard version of that language, on the way making life much easier for the programmer by handling nice features like macro expansion, and nice control structures like "while" and "repeat". Generally a preprocessor program is less efficient with respect to speed and space than a program written "manually", but this is generally considered a minor tradeoff for the greater ease and speed of program development -- this is why most people don't program in machine code when they can use a high level language which will do just as well.

The other day, as I was sitting at my computer wondering what to do next, when a rabbit ran past, looking at its watch..No, we don't have any rabbits where I live. I was struck by an idea that Ratfor was a boon to Fortran; how about something similar for BASIC, another unstructured language? I left for the pen and paper, and a few hours later the Ratbas program was running.

A Ratbas program is essentially a BASIC program with some extra features:

if(<condition>)	if(<condition>)
<statements>	<statements>
endif	else
	<statements>
	endif
while(condition)	repeat
<statement>	<statement>
endwhile	until(<condition>)
switch(<expression>)	
case(<expression>)	
<statement>	
case(<expression>)	
<statement>	
...	--any number of cases are
	allowed.
default	--this is not optional within
<statement>	a switch statement.
endswitch	
break	continue
--break out of a	--jump to the condition test
while, repeat or	of the innermost while or
switch statement	repeat statement.
subr"<name> ---	call"<name>
"<name> ---	goto"<name>

The Ratbas extensions enable the writing of a program without a single line number being referenced; this makes things much nicer to both write and read. A Ratbas program is typed in and edited, loaded and saved, in BASIC, just like a standard BASIC program; it just won't run until it is processed.

The Ratbas preprocessor is invoked in the following manner:
run"ratexec"

This program asks for the name of the Ratbas program to be processed. It sets up a nifty device #5 command list which loads the program to memory, then lists it to a text file, then invokes the program "Bigrat" which then reads the text file, identifying Ratlines and outputting the appropriate code or each such line to a second text file. If no errors are encountered, another device #5 command list is set up which loads the original text file, then overwrites the Ratlines with the processed lines from the second text file; and then the processed BASIC program is sitting in memory, ready to be run or saved.

The Bigrat program was written in BASIC, and so the Ratbas statement syntax has been tailored for efficiency. A Ratbas statement must sit on a line of its own, prefixed with a quote. Normal BASIC lines don't need to follow these restrictions. The lines in a program can be indented with spaces, giving a nice tiered effect to the nested loops and statements -- yes, the control structures can be nested, to a depth of 100, which seems to be (far) more than adequate for any properly written program. The sample program below should demonstrate the uses of the above constructs.

```

100 REM <GUESS.RAT> --Sample Ratty program
110 PRINT"Think of a number between 1 and 100."
120 PRINT"I will try to guess it."
200 low=1:high=100
210 true=-1:false=0
220 found=false:REM Ie. haven't guessed the number yet.
250 "repeat
300 : guess=INT((low+high)/2)
310 : PRINT"Is it (L)ower or (H)igher than, or (E)qual to"guess;
320 : INPUTa$
330 " switch(left$(a$,1))
340 " case("L")
350 : high=guess-1:REM Try lower next time.
360 " break
370 " case("H")
380 : low=guess+1:REM Try higher next time.
390 " break
400 " case("E")
410 : PRINT"Hooray! I guessed it!":REM Nauseating, but apt.
420 : found=true
430 " break
440 " default
450 : PRINT"That's not an appropriate answer!"
460 " break
540 " endswitch
550 "until(found)
560 END
570 "endpr

```

THIS IS WHAT THE
PROCESSED PROGRAM
LOOKS LIKE.

```

100 REM <GUESS.RAT> --Sample Ratty program
110 PRINT"Think of a number between 1 and 100."
120 PRINT"I will try to guess it."
200 low=1:high=100
210 true=-1:false=0
220 found=false:REM Ie. haven't guessed the number yet.
250 REM
300 : guess=INT((low+high)/2)
310 : PRINT"Is it (L)ower or (H)igher than, or (E)qual to"guess;
320 : INPUTa$
330 REM
340 IFLEFT$(a$,1)<>("L")THEN 370
350 : high=guess-1:REM Try lower next time.
360 GOTO 540
370 IFLEFT$(a$,1)<>("H")THEN 400
380 : low=guess+1:REM Try higher next time.
390 GOTO 540
400 IFLEFT$(a$,1)<>("E")THEN 440
410 : PRINT"Hooray! I guessed it!":REM Nauseating, but apt.
420 : found=true
430 GOTO 540
440 REM
450 : PRINT"That's not an appropriate answer!"
460 GOTO 540
540 REM
550 IFNOT(found)THEN 250
551 REM
560 END
570 "endpr

```

Ok
run

- AND IT EVEN WORKS!

```

Think of a number between 1 and 100.
I will try to guess it.
Is it (L)ower or (H)igher than, or (E)qual to 50 ? l
Is it (L)ower or (H)igher than, or (E)qual to 25 ? zooble
That's not an appropriate answer!
Is it (L)ower or (H)igher than, or (E)qual to 25 ? h
Is it (L)ower or (H)igher than, or (E)qual to 37 ? h
Is it (L)ower or (H)igher than, or (E)qual to 43 ? l
Is it (L)ower or (H)igher than, or (E)qual to 40 ? h
Is it (L)ower or (H)igher than, or (E)qual to 41 ? h
Is it (L)ower or (H)igher than, or (E)qual to 42 ? e
Hooray! I guessed it!

```

A listing of the program will be printed in next month's KAOS but if you don't feel like typing it in Michael will be making the program available on disk, through Comp-Soft, for \$10 plus P/P.

A CORRECTION TO ROM BASIC

by Rodney Eisfelder

After many hours of hard thinking, the solution to another bug in ROM BASIC can now be revealed. The problem occurs in systems with more than 8K of RAM. When an INPUT statement is between \$2000 and \$20FF then the first character typed in response to the INPUT is ignored as well as the first non-space character. The problem is described in the 'Dear Paul' column in KAOS 3.6.

The solution is to change two bytes of the second BASIC ROM. The two bytes are \$A969 and \$A9CD (or in English, 43369 and 43469) which currently have the value \$12. This is the address used to save the high byte of the BASIC program counter and is also immediately before the BASIC line input buffer.

The problem occurs when the high byte of the BASIC program counter is the same as an ASCII space i.e. \$20. When BASIC starts processing an INPUT line, the buffer pointer points one byte before the start of the buffer, that is it points to \$12. The get-current-character routine (\$00C2) is called to detect end of line. For the first INPUT variable this is not meant to do anything because a special test is made for zero length INPUT lines. However, if the byte before the buffer is a space, then the pointer will be moved on and the first character skipped. BASIC even goes to the trouble of writing \$2C (a comma) into \$12 before overwriting it with the program counter.

The solution is therefore to change the two bytes mentioned above so that a harmless location is used to save the BASIC Program Counter. Any location not used elsewhere by BASIC is obviously 'safe' to use and I would suggest \$D8 as a contender. People who suffer from this problem should note that my solution is not tested and is therefore as reliable, complete and correct as any untested program can be.



XMAS SPECIALS FROM COMP-SOFT

Whatever-80 printers still on special at \$399.00 inc. tax
Apple printer interface for the above \$95.00 inc. tax
8" slimline disk drives still \$535 inc. tax
Box of paper a steal at \$29.00 inc. tax
Microline 82A printer \$695.00 inc. tax
Case 515 132 col. printer \$895.00 inc. tax
Tasan video boards (bare) \$20.00 inc. tax
AMDEK high resolution green screens \$225.00 inc. tax
Peach computer with 32K ram, dual S/S disk drives, controller etc.
\$1695.00 inc. tax

NEW.....

5 1/4" slimline 40 track single sided disk drive -
The F-051 SHINON disk drive is a precision made drive produced by a Japanese company who specialise in cameras.
Price for the SHINON disk drive is \$295.00 inc. tax
Case and power supply for the above drive is \$95.00 inc. tax
EPROM BURNER BOARD (bare) \$18.00 inc. tax
Power Supply board for EPROM board (bare) \$6.00 inc. tax
Ring for kit price

COMP-SOFT, 235 Swan St, Richmond, 3121. PH. 03 429 9686, 03 428 5269

MORE MODS TO THE EXTENDED MONITOR
by Rodney Eisfelder

Anyone who has attempted to use the break point facility of the extended monitor to debug machine code programs will have discovered that one of the biggest problems is remembering what breakpoints you have already used and where they are.

Here is the solution to that problem. The J command displays a table showing where all eight breakpoints are and what the replaced byte is.

```

100      ;
110      ; 'J' Command for Extended Monitor
120      ; (Disk Version - Cassette Version should be
130      ; similar, but at a different address)
140      ;
150      ; Prints Table of Breakpoint addresses
160      ;
170      ; Written by R.Eisfelder Nov 1983
180      ;
190      ; Don't forget to modify command table.
200      ; $18B3/B4 Changes from $96/$17 to $97/$1F
210      ;
220 1BAF=    SPACE = $1BAF ;Print space
230 19E9=    TWOHEX = $19E9 ;Print A as 2 Hex Chars
240 1A56=    CRLF = $1A56 ;Print Return,Line feed
250      ;
260      ; Start at first free address after adding
270      ; modifications to search command (KAOS 2.12)
280      ; and Z command (KAOS 3.2 & Correction 3.3)
290      ; relocated to $1F53-$1F96)
300 1F97      * = $1F97
310 1F97 A001      LDY #1
320 1F99 A200      LDX #0
330 1F9B 98      LOOP    TYA
340 1F9C 09B0      ORA #$B0
350 1F9E 20E919      JSR TWOHEX ;Print Bn where n
360                      ;is the breakpoint number
370 1FA1 B5F1      LDA $F1,X
380 1FA3 20E919      JSR TWOHEX ;Print Address High
390 1FA6 B5F0      LDA $F0,X
400 1FA8 20E919      JSR TWOHEX ;Print Address Low
410 1FAB 20AF1B      JSR SPACE
420 1FAE B9D700      LDA $D7,Y
430 1FB1 20E919      JSR TWOHEX ;Print Value replaced by BRK
440 1FB4 20561A      JSR CRLF
450 1FB7 E8      INX
460 1FB8 E8      INX
470 1FB9 C8      INY
480 1FBA C009      CPY #9
490 1FBC 30DD      BMI LOOP
500 1FBE 60      RTS

```

by John Whitehead

To enable me to program 2764 EPROMS and fit 4 of them onto a modified Tasker EPROM board, I have built another programmer, complete with power supply.

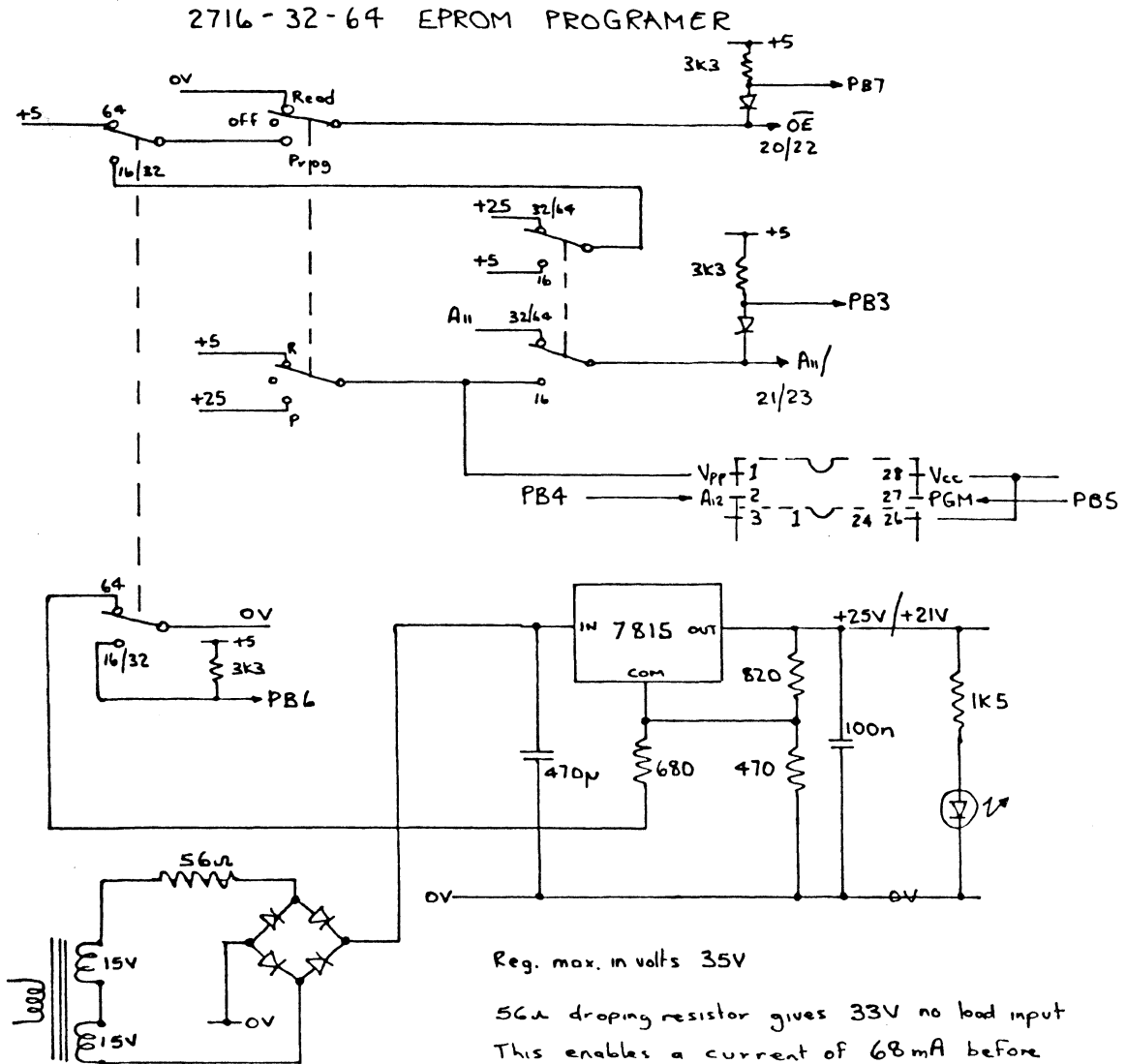
A 2764 need 21 volts to program it and 2716's and 2732's need 25 volts. The programmer connects to a PIA.

All switch positions are detected except between OFF and PROGRAM. This has not caused me any problems as the software displays every switch movement required.

The circuit has been built on veroboard. To keep 240V off the main board the transformer was mounted on a separate bit of board spaced away from the other with veropins. The Arlec AL7VA/30 transformer was one I already had and as its output voltage was too high for the 7815 regulator, I dropped it down with a 56R resistor. The LED must be included as a permanent load.

The software is a modified version of my 2716-32 BASIC program (KAOS Dec 82) and either program is available on tape for \$2 and \$1 postage.

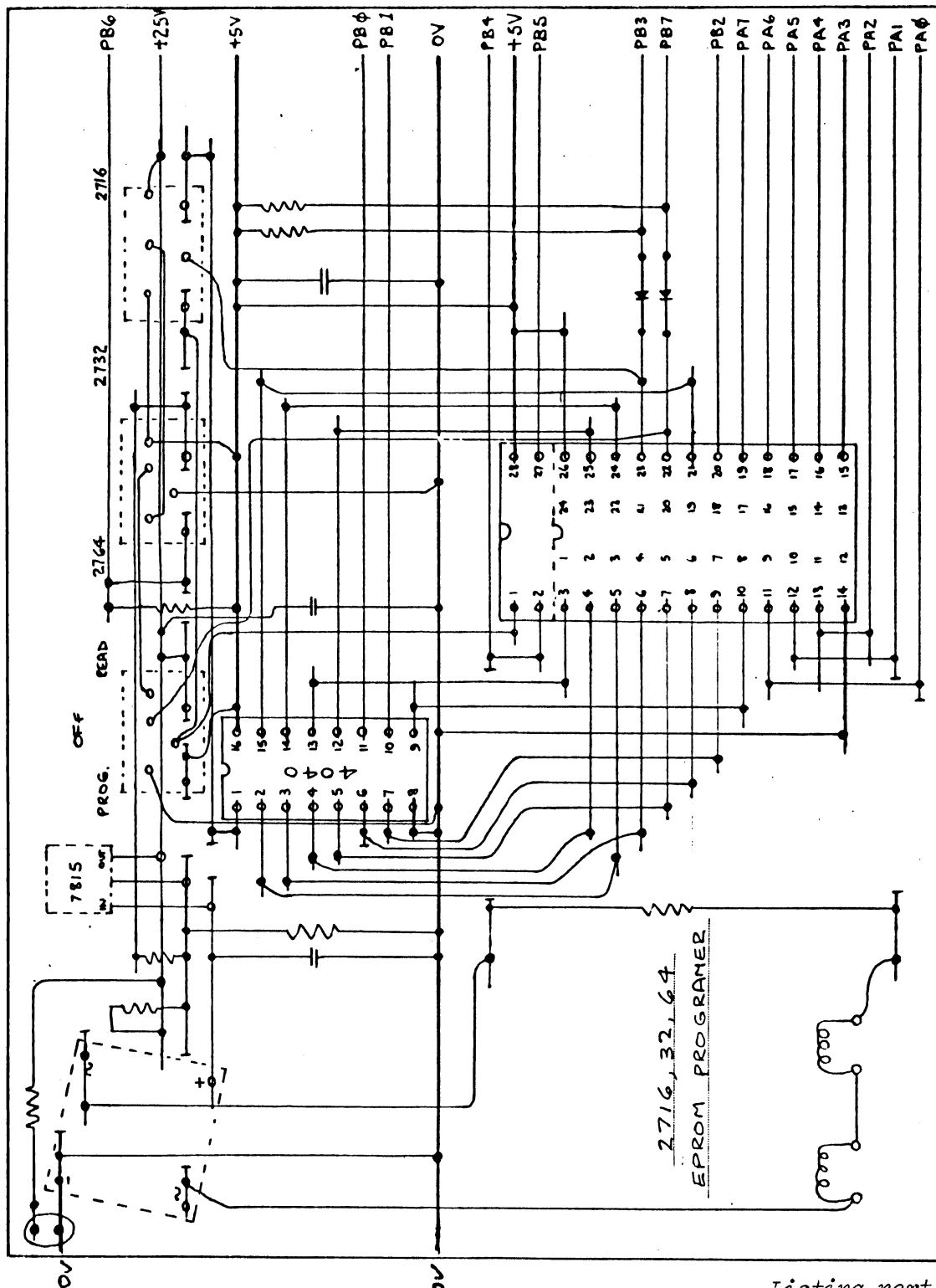
So far I have a Word Processor in one 2764 and BASIC utilities in another.



Reg. max. in volts 35V

56 Ω dropping resistor gives 33V no load input
This enables a current of 68mA before
reg. output voltage drops.

Input voltage is measured between IN and COM.



Listing next page

40 PIN SOCKET EXPANSION

Bert Patterson has made up a 40 pin socket extender board. It has 4 extra 40 pin sockets on it, and plugs into the 40 pin J1 socket on the C1P.

All lines except the data lines (already buffered on the C1P) are fully buffered. It also has a DD line to suit the Elector Disk controller board, but this may be left unpopulated or the socket used as a proto area.

The board is available from him, 20 Donegal St, Salisbury Downs, SA 5108 for \$15 undrilled or \$20 drilled plus \$2 for P&P.

```

:-----:
5 :
6 REM  EPROM burning program by John Whitehead
7 :
8 :-----:
9 :
10 PRINTCHR$(127)"  EPROM burning routine J060883W11
20 PRINT:PRINT"  Set for 2MHz & a PIA at $C004-7":PRINT
30 PIA=49156:GOSUB 1110:REM A to inputs
40 POKEPIA+3,0:REM  Set B0,1,2,4,5 outputs
50 POKEPIA+2,55:POKEPIA+3,4
60 :
70 REM  Insert
80 :
90 R=0:GOSUB1190:REM Set B outputs zero & reset address
100 P=PEEK(PIA+2):PRINTCHR$(13)"Switch to OFF";:IFP<128THEN100
110 P=PAND127:IFP=8THENEP=2716:K=2048
120 IFP=72THENPRINT"Incorrect selection":GOTO90
130 IFP=0THENEP=2732:K=4096
140 PRINT:IFP=64THENEP=2764:K=8192
150 PRINT:PRINT"  Insert a"EP"EPROM. Then switch to READ
160 PRINT"  -----
170 P=PEEK(PIA+2):IFP>127THEN170
180 :
190 REM  Get address
200 :
210 PRINT:INPUT"EPROM base address in hex ";A$:GOSUB 1140:ES=C
220 IFES/K<>INT(ES/K)THENPRINT" BASE ERROR":GOTO210
230 INPUT"Start burning from address ";A$:GOSUB 1140:SB=C
240 INPUT"End burning at address ";A$:GOSUB 1140
250 PRINT:IFC<SBORSB<ESTHENPRINT" Error":GOTO210
260 IFC<ES>K-1THENPRINT"Size error":GOTO210
270 PRINT"EPROM base address"ES
280 PRINT"Data from memory at address"SB"to"C
290 :
300 REM  Menu
310 :
320 PRINT:PRINT"Test for a clear EPROM = T
330 R=32:GOSUB1190:PRINT"Program and compare  = P
340 A=ES:PRINT"Compare  = C
350 PRINT"Move from EPROM to RAM = M
360 INPUT"Reset PIA and END  = R ";Z$:IFZ$="P"THEN540
370 IFZ$="C"THEN830
380 IFZ$="M"THEN950
390 IFZ$="R"THEN1030
400 :
410 REM  Clear test
420 :
430 F=0:PRINT" Clear testing":FORA=ESTOC
440 IFA<ES=4096THENR=48:GOSUB1190:REM Set A12 high
450 IFA=>SBTHEND=PEEK(PIA):IFD<>255THENF=F+1:PRINTA,D,
460 POKEPIA+2,R+2:REM Increment address
470 POKEPIA+2,R
480 NEXT:PRINT:PRINT"locations are not FF
490 IFF<>0THENPRINT" Its not clear
500 GOTO320
510 :
520 REM  Programming
530 :
540 GOSUB 1080:REM  A to output
550 IFEP=2716THENR=0:T=136:BURN=4
560 IFEP=2732THENR=4:T=132:BURN=0
570 IFEP=2764THENR=32:T=224:BURN=0
580 POKEPIA+2,R:REM Reset burn pulse
590 :
600 PRINT:PRINT"Ensure +25V is on and switch to PROGRAM a"EP
610 IFPEEK(PIA+2)<>TTHENPRINTCHR$(13)"Switch to 2MHz";:GOTO610
620 PRINT:FORX=0TOEP:NEXT:REM Switch debounce
630 PRINT:PRINT"Programing a"EP"in progress
635 IFEP=2716THEN670
640 IFA<ES=4096THENR=48:BURN=16:GOSUB1190:REM  A12 HIGH
650 IFA<ES=2047ORA<ES=6143THENIF(PEEK(PIA+2)AND8)<>0THENSTOP
660 IFA<ES=2048ORA<ES=6144THENIF(PEEK(PIA+2)AND8)<>0THENSTOP
670 IFA<SBTHEN740:REM Inc. address until =
680 D=PEEK(A):REM  fetch data
690 POKEPIA,D:REM Load data
700 :
710 POKEPIA+2,BURN
720 FORX=0TO100:NEXT:REM Hold for 50ms
730 POKEPIA+2,R:REM BURN OFF
740 POKEPIA+2,R+2:POKEPIA+2,R:REM Inc address
750 IFA<CTHENA=A+1:GOTO635
760 :
770 PRINT"Programing finished.
780 IFPEEK(PIA+2)>127THENPRINTCHR$(13)"Switch to READ";:GOTO780
790 PRINT:FORX=0TOEP:NEXT:GOSUB1110:R=32:GOSUB1190
800 :
810 REM  Compare
820 :
830 PRINT"Comparing in progress
840 F=0:PRINT:FORA=ESTOC:D=MD
850 IFA<ES=4096THENR=48:GOSUB1190:REM Set A12 high
860 IFA=>SBTHEND=PEEK(PIA):MD=PEEK(A)
870 IFD<>MDTHENF=F+1:GOSUB1230
880 POKEPIA+2,R+2:POKEPIA+2,R:REM Inc address
890 NEXT:PRINT"locations are not the same
900 IFF<>0THENPRINT" Its not programed correctly
910 GOTO320
920 :
930 REM  Move
940 :
950 PRINT:PRINT"Moving in progress":FORA=ESTOC
960 IFA<ES=4096THENR=48:GOSUB1190:REM A12 HIGH
970 IFA=>SBTHEND=PEEK(PIA):POKEA,D
980 POKEPIA+2,R+2:POKEPIA+2,R:NEXT:REM Inc ad
990 PRINT:PRINT"  finished":PRINT:R=32:GOSUB1190:GOTO830
1000 :
1010 REM  Remove
1020 :
1030 IFPEEK(PIA+2)<128THENPRINTCHR$(13)"Switch to off";:GOTO1030
1040 R=0:GOSUB1190:PRINTCHR$(127)"  Remove EPROM":END
1050 :
1060 REM  Subroutines
1070 :
1080 POKEPIA+1,0:REM A all outputs
1090 POKEPIA,255:POKEPIA+1,4:RETURN
1100 :
1110 POKEPIA+1,0:REM A all inputs
1120 POKEPIA,0:POKEPIA+1,4:RETURN
1130 :
1140 C=0:FORX=1TO4:E=ASC(MID$(A$,X,1))
1150 IFE>70THENPRINT" Error":GOTO1190
1160 E=E-48:IFE>16THENE=E-7
1170 C=C*16+E:NEXTX:RETURN
1180 :
1190 POKEPIA+2,R:REM B outputs to zero if R=0
1200 POKEPIA+2,R+1:REM Reset address
1210 POKEPIA+2,R:RETURN
1220 :
1230 H8=A:GOSUB1260:PRINT"At "A$;:H=1
1240 H8=MD:GOSUB1260:PRINT" RAM = "A$;
1250 H8=D:GOSUB1260:PRINT" EPROM = "A$;H=0:RETURN
1260 H5=H8/4096:H4=INT(H5):H6=(H5-H4)*16
1270 H3=INT(H6):H7=(H6-H3)*16:H2=INT(H7):H1=INT((H7-H2)*16+.5)
1280 IFH1>9THENH1=H1+7
1290 IFH2>9THENH2=H2+7
1300 IFH3>9THENH3=H3+7
1310 IFH4>9THENH4=H4+7
1320 A$=CHR$(H2+48)+CHR$(H1+48):IFH=1THENRETURN
1330 A$=CHR$(H4+48)+CHR$(H3+48)+A$:RETURN
1340 :
1350 REM change FOR-NEXT in line 720 to 50 for 1MHz
OK

```

BASIC QUIRK

by George Nikolaidis

One of the quirks of Ohio Microsoft BASIC has been the fact that it refused to accept line numbers greater than 63999. It seems logical that if BASIC uses a 16 bit number then you should be able to get numbers up to 65535.

I recently discovered the reason for this. The 'bug' lies in the ASCII to hex number routine in BASIC. This is a simple routine which uses a very crude method to detect if the number is too large. It compares the high byte of the number \$19XX which is 64000 just before its multiplied by 10. This is the simplest way for them to check if the number is too large. A better way can be used to achieve this but I'll leave that up to you.

TAPE VIEWER
from O.S.U.G. newsletter 6.

Here is a short routine you call with the USR function. It prints on the screen exactly as if you were loading a Program. You can use Cursor Control to put all or part of the listing into your current program. To leave the routine you press the space bar as you would normally stop a LOAD. The program occupies the top of the Stack, and should be safe from erasure unless there is 10 levels of brackets in your resident program. A cold start does not wipe it out.

0131 A9 FD 8D 00 DF A9 10 2C 00 DF F0 09 20 80 FE 20 2D BF 4C 31 01 60

Set up the USR call with: POKE11,49:POKE12,1

Call at any time with: X=USR(X)

For a C2/C4, change: 0132=02 013B=D0 013E=07 013F=BF



FOR SALE

A.L.F.O. - the 'Alternative language for Ohios' is now available for 8K C1 or C4 computers. It is an educational and interesting language - great for people bout to learn Assembler. See November KAOS for a review of ALFO. For the program (including P&P and 33 pages of documentation) send \$9.95 to Brad Monsbourgh,

AIM 65 System. Comprises of 4K AIM 65 with Assembler & Basic in ROM, 32K dynamic RAM expansion, power supply with case and fan, 5 Rockwell manuals & Basic cheat card. \$750.00
(ask for Dieter)

A program which converts any ASCII string to speech using the SC-01 Synthetiser Chip.
Available on 5 1/4" disk or cassette.
Price \$20 from Paul Brodie,

Original BASIC 5 ROM plus documentation. Needs Cegmon.
Price \$15 from Ken McNeill,

Tasker Bus. 8 slot motherboard plus 40 pin header card plus 3 x 8K RAM cards, all fully populated.
Price from Paul Brodie,

SUPERBOARD Series 1, in case with power supply, fan, 32K RAM, floppy disk controller, plus converted T.V.
Price \$300 Contact Russell Cooper

Registered by Australia Post
Publication No. VBG4212

If undeliverable, return to
KAOS, 10 Forbes St
Essendon, Victoria 3040

KAOSKAOS
K Postage K
A Paid A
O Essendon O
S 3040 S
KAOSKAOS